From the Editor in Chief...

Air System Information Management

Robert E. Filman • *RIACS/NASA Ames Research Center* • *filman@computer.org*

I flew to Washington, DC, last week — a trip rich in distributed information management. Buying tickets, at the gate, in flight, landing, and at the baggage claim, myriad messages about my reservation, the weather, our flight plans, gates, bags, and so forth flew among a variety of travel agency, airline, and US Federal Aviation Administration (FAA) computers and personnel. By and large, each kind of information ran on a particular application, often specialized to its own data formats and communications network.

I went to Washington to attend an FAA meeting on System-Wide Information Management (SWIM) for the National Airspace System (www.nas-architecture.faa.gov/Tutorials/NAS101.cfm). NAS (and its information infrastructure, SWIM) is an attempt to bring greater regularity, efficiency, and uniformity to the collection of stovepipe applications now used to manage air traffic. Current systems hold information about flight plans, flight trajectories, weather, air turbulence, current and forecast weather, radar summaries, hazardous condition warnings, airport and airspace capacity constraints, temporary flight restrictions, and so forth.

Information moving among these stovepipe systems is usually mediated by people (for example, air traffic controllers) or single-purpose applications. People, whose intelligence is critical for difficult tasks and unusual circumstances, are not as efficient as computers for tasks that can be automated. Better information sharing can lead to higher system capacity, more efficient utilization, and safer operations. Better information sharing through greater automation is possible, though not necessarily easy.

An Airspace Utopia

Ideally, when an airplane flying over the Rocky Mountains encounters turbulence and shifts its course, all "interested" parties could be made aware of the situation. For example, pilots of other airplanes on that flight path could be warned to avoid the turbulence; workers at the baggage service in San Francisco, having a better idea of the real arrival times, could reschedule the luggage carousels; operators of connections for the passengers on that flight could make appropriate decisions about holding (or not holding) and routing flights; air traffic controllers could verify that the plane's new trajectory doesn't collide with some other craft; climate researchers on long-term turbulence patterns could add that incident to their database; and countless other future applications that might care about airspace information could patch into the information network.

What do we demand of a system moving that much flight information? Important "-ilities" include

- efficiency (you can't tell everyone everything),
- evolvability (you don't know all the future applications),
- scalability (this is a big system, it will grow, and you don't want it to be architecturally limited in capacity or vulnerable to a few points of failure),
- maintainability (when things don't work, you must be able to quickly find out why),
- reliability (for obvious reasons),
- quality of service (you need to get important information to its destinations quickly, deferring the unimportant — information about a hijacking takes priority over climate research, for example) and
- security (information should be injected into and removed from the system only by appropriate people).

The definition of appropriate in this last point includes not only keeping black hats from injecting spurious information into the system, but also the commercial concerns about sharing business information. An airline is willing to tell traffic control that a plane is late. It might even be willing to share that information with its own ground operations. However, the airline will often want to keep that information not only from its competitors, but

 Published by the IEEE Computer Society 1089-7801/04/$20.00

also from its own gate personnel. (This saves the clerk at the gate from having to conceal a lie when trying to convince you not to switch to another airline just because your flight won't take off for another three hours.)

Foundations of an Airspace Utopia

The desire to inform "all interested parties" argues for a publish-and-subscribe architecture. In publish-and-subscribe, *publishers* generate *events* in the course of program execution. In our airspace example, the flight over the Rockies would publish events announcing the turbulence and its new routing. *Subscribers* present to the system *subscriptions*: descriptions of the kinds of events that interest them. When the system detects events that match subscriptions, it notifies the subscribers with those events.

There is a lot of design space within the scope of publish-and-subscribe. The first important decision relates to the structure of events. Possible structures for events include record-like, free-form, tree-structured (such as XML), and objects. Concomitant with the structure of events are their guaranteed properties. For example, we can demand that every event be signed, time-stamped, and marked for expiration. We might wish to distinguish among the information in the event envelope (for example, its generator), its content, and annotations about it (such as when or where it happened or its accuracy or precision).

Subscriptions talk about something and provide various operators for describing what makes events interesting. The simplest subscription mechanism is one based purely on the event's type — for example, a subscription to all turbulence events. While type-based subscriptions might work for controlling a workstation user interface, the distribution and complexity of the airspace system demands a more refined approach. A better subscription language allows describing a predicate over the envelope, content, and annotations of a single event — for example, all events whose "type" is "turbulence," whose location is within 300 miles of Denver, and which happened on 19 January.

The volume of information circulating in a national air system could demand a richer notion of subscription. We might find ourselves wanting to subscribe to a sequence of events that match some temporal property (for example, "five or more reports of late takeoffs from an airport within an hour"), that select from the set of generated events (for example, "the location of Flight 007 every three minutes"), or even to embed programs to winnow through events within (agent-oriented) subscriptions. Subscriptions might express choreography for the results or

have their effects limited by the subscriber's access privileges. There is a tension between efficiency and richness of expression in the subscription language: we want to allow applications to precisely specify what information they want, but also want to minimize the work involved in culling that information for them.

Communication relies on shared structure and shared understanding. (It also relies on actual communication channels, the efficient maintenance of which is challenging for communicants moving at 1,000 kilometers/hour. However, concrete communication is beyond the scope of this abstract discussion.) Part of developing such a system includes making decisions on common data formats (such as XML) and common data meanings (such as ontologies for airspace management). Thus, a subscription to turbulence events is likely to refer to an ontology of wind conditions. A clever system might even provide automatic or semiautomatic mechanisms for translating between representations, much as a client that generated a subscription to aircraft location in terms of Pacific Standard Time might see an automatic transformation of events that are really generated in Universal Time. Ontologies are important; even within the FAA discussions, it was amusing to note the different uses of the term "surveillance" — air traffic controllers use it with respect to knowing where an airplane is; security folk want to watch what's happening on the plane. Similarly, for an application so concerned with where things are, a common notion of and expression for geography and trajectory could prove critical.

If subscribers can retrieve events that happened before their subscription was entered, the system will need a mechanism for storing past events, matching them against new subscriptions, and forwarding the results. Storage issues also get tangled with legal requirements for data preservation and access.

An architecture based on bringing all events to a single locus for sorting against all subscriptions is unlikely to scale to the national airspace or provide the necessary reliability to actually fly planes. Thus, a major issue in such a publish-and-subscribe system is the organization and replication of *brokers* (that is, event channels) that mediate between publishers and subscribers. Because the system must grow to encompass new applications, discovery and allocation are critical issues: matching brokers to the kinds of messages they receive and republish. We might find a need for a hierarchical broker topology, the ability for brokers to forward subscriptions (or parts of subscriptions) to each other, and a facility that lets a forwarding broker collate multiple answers to a single form.

In airspace control, the same information often has several sources. For example, you can find out where an aircraft is by projecting from its original flight plan, reading a current radar trace, or extrapolating the aircraft's message about its GPS location and trajectory. The underlying architecture might not need to arbitrate between information sources. However, thinking about how to organize with respect to multiple information sources is a

critical architectural issue.

We have been discussing communication abstractions. Of course, building a real airspace control system involves concretely handling the issues of real systems:

- performing all facets of security, including authentication, intrusion detection, and access control;
- providing reliability to a life-critical system;
- meeting applications' real-time and data-streaming needs;
- accounting for the consumption of resources;
- monitoring performance;
- enabling configuration management; and
- identifying faults.

Even if an ideal solution to all these problems emerged, Venus-like, whole and complete, an even larger issue is how to achieve a smooth and reliable transition from current practice. Airspace management is a working, life-critical, heterogeneous, real-time, always-running application used not only by large, well-identified client organizations (such as airlines) but also by individuals (for example, private pilots in noncommercial aircraft). It can't be changed instantaneously; it can't be turned off to bring up a new version. It must work, and there are many stakeholders with differing opinions about the value and desirability of improvements.

Airspace Information and the Internet

We already have a distributed information system in the Internet. How well do the Internet, TCP/IP, DNS, and their kin stack up against the requirements of airspace information management? A great virtue of the Internet is its arbitrary connectivity – any process can potentially communicate with any service. Arbitrary connectivity's importance for unexpected evolution cannot be exaggerated: traditional system

IC Welcomes New Editorial Board Member

Monika Henzinger is the research director at Google.

Prior to joining Google, she was a research staff member at Digital's Systems Research Center in Palo Alto, Calif., where she worked on Web information retrieval and systems performance measurements as well as graph algorithms and data structures, which are useful for mapping the Web's topography and behavior.

Henzinger received a PhD from Princeton University in 1993. She then joined the Computer Science Department at Cornell University as an assistant professor and received a US National Science Foundation Career award in 1995. Henzinger is also a recipient of the Wallace Memorial Honorific Fellowship at Princeton University, a Studienstiftung des deutschen Volkes fellowship, and a Siemens Scholarship fellowship.

development for applications like air-traffic controls starts by drawing boxes connected by lines, and is flummoxed when new applications demand additional connections. It's also important to keep in mind the lessons of the end-to-end argument: put the intelligence in the applications, not the network. This universality must be balanced by the knowledge that we are discussing a system for airspace information management, not unstructured communication.

That said, the current Internet architecture doesn't match our application in a variety of ways. The poor quality of IPv4 Internet security (which IPv6 might ameliorate), the lack of real-time capabilities, the difficulty in managing the Net as a whole, the lack of inherent accounting mechanisms, and the limitations on network evolvability should be familiar to our readers. More interesting as an abstraction issue is the philosophical shift from addressed to content-directed communication. It will be interesting to see how radically we'll need to change our thinking about networks to accomplish these goals.

Closing Remarks

The theme of the Communications Abstractions workshop at ECOOP in Oslo this June will be "communication abstractions for airspace management." Please submit a position paper if you'd like to come (http://perso-info.enst-retagne.fr/~beugnard/ecoop/WS-CADS04-CFP.html). My thanks to Steve Bradford, Brian Glass, Josh Hung, and Jack Levine for comments on the drafts of this column. My work on technologies for SWIM is supported by NASA's Airspace Program.